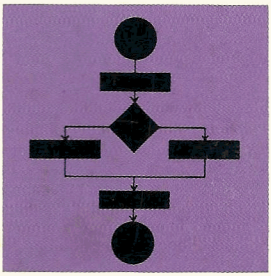
THE PROGRAMMING BOOK

A GUIDE TO PROGRAMMABLE CALCULATORS

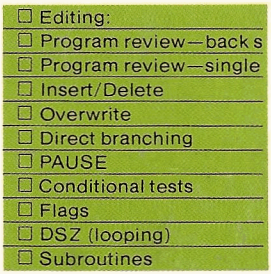
CONTINUOUS MEMORY

FULLY-PROGRAMMABLE

KEYCODES



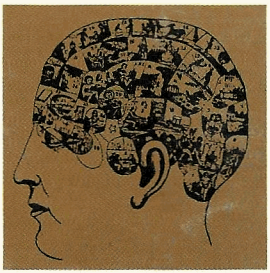
FLOWCHARTING



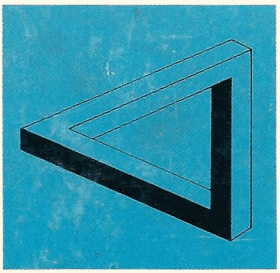
WHAT YOU NEED

POWER

TIME-SAVING



PROGRAM MEMORY



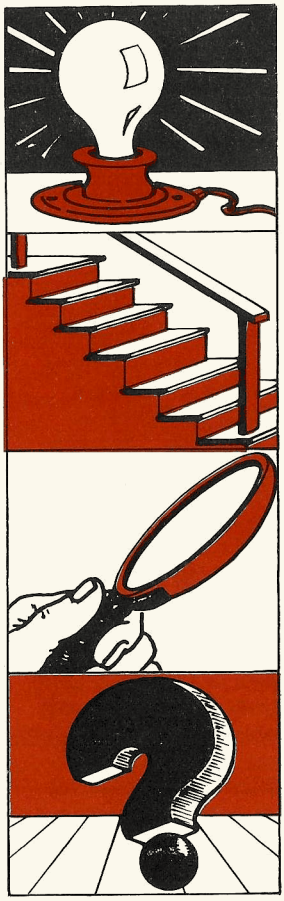
LOGIC SYSTEM

HEWLETT hp PACKARD

Foreword

In the past few years an unprecedented computational capability has been made available to anyone who needs it in the form of the pocket programmable calculator. Like the large scale computers from which they are descended, these miniature marvels can be programmed by their users to solve an amazing variety of complex or repetitious calculations in virtually any field.

This booklet is designed to acquaint you with the versatility and power of pocket programmable calculators—while removing any doubts or apprehensions you may have. And if you decide to join the rapidly growing numbers of programmable calculator users, this booklet contains an assortment of hints to help you select the machine which best meets your requirements.



Contents

2 Programming–
An Enlightened Approach
to Problem Solving

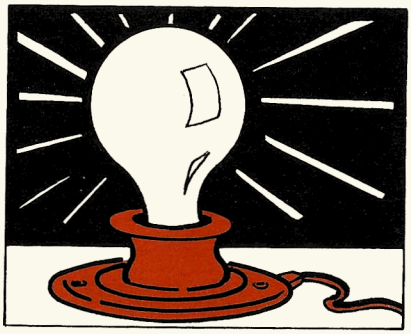
3 Seven Steps for Writing
and Using Effective Programs

7 Programming Features
You Should Know About

13 Choosing a
Programmable Calculator:
What You Need To Know

15 A Final Note

Programming– An Enlightened Approach to Problem Solving



The concept of programming is actually as old as finger counting and the abacus. In fact, *everyone* uses mental "programs" to make decisions, solve problems and, of course, operate calculators.

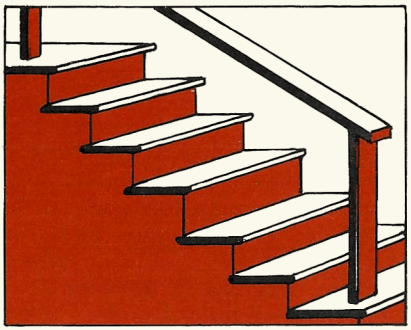
The pocket programmable calculator combines the capabilities of advanced scientific calculators with a built-in ability to *remember* the sequence of keystrokes necessary to solve a particular problem. You simply key in the problem once. The calculator will remember and execute the identical keystrokes each time you press one or two instruction keys.

Programmable calculators can have many advanced features, some of which we'll examine in this booklet. But their most important attribute is that you can learn to use even the most sophisticated programmable calculator with no prior programming experience. You simply switch the calculator to its *program* or *learn* mode, key in the problem exactly as you would when solving it on a non-programmable machine, and switch the calculator back to its *run* mode. Then you key in the appropriate data and execute the problem as many times as you wish. The calculator takes care of the rest.

In addition to programs you write yourself, you can choose from hundreds of programs available at very low cost from some of the manufacturers of programmable calculators. These programs will allow you to solve complex problems easily in such fields as business, statistics, mathematics, medicine, engineering and physics by simply keying in the appropriate program.

Of course calculator manufacturers cannot possibly anticipate everyone's problem-solving requirements, so there will still be times when you need to develop your own programs.

Seven Steps for Writing and Using Effective Programs



Keystroke programmability means you can often key simple programs directly into a programmable calculator without advance planning. Complex problems, however, require a more organized approach to programming. A simple seven-step method for writing and using effective programs follows.

1. Identify the solution you require and the information on hand.

Before you can write a program you must first identify the result you are seeking—and the information on hand which will help you reach it. For example, let's assume Elmer Travelmore is planning a summer vacation to one of three scenic beaches. If Elmer's vintage roadster averages 22 miles per gallon and gas costs 61 cents per gallon, what will his round-trip gasoline expense be for destinations 226, 314, and 289 miles distant?

Obviously, the solution required in this problem is the total gasoline expense for each of three proposed trips. The information on hand is the rate of gasoline usage, the cost of a gallon of gasoline, and the distance to each of three destinations.

2. Devise a plan of attack.

After you identify the information on hand and the solution you require, you can convert the problem into a mathematical expression. In the case of our example problem, the distance to each destination must first be doubled to find the total round-trip mileage. Then each mileage figure must be divided by the number of miles Elmer's car travels on a gallon of gas to find the quantity of gasoline. Finally, the quantity of gasoline must be multiplied by the cost per gallon.

These steps can be condensed into the following statement:

Distance in miles x 2 (round trip) ÷ miles per gallon (22) x cost per gallon ($.60) = gasoline expense ($).

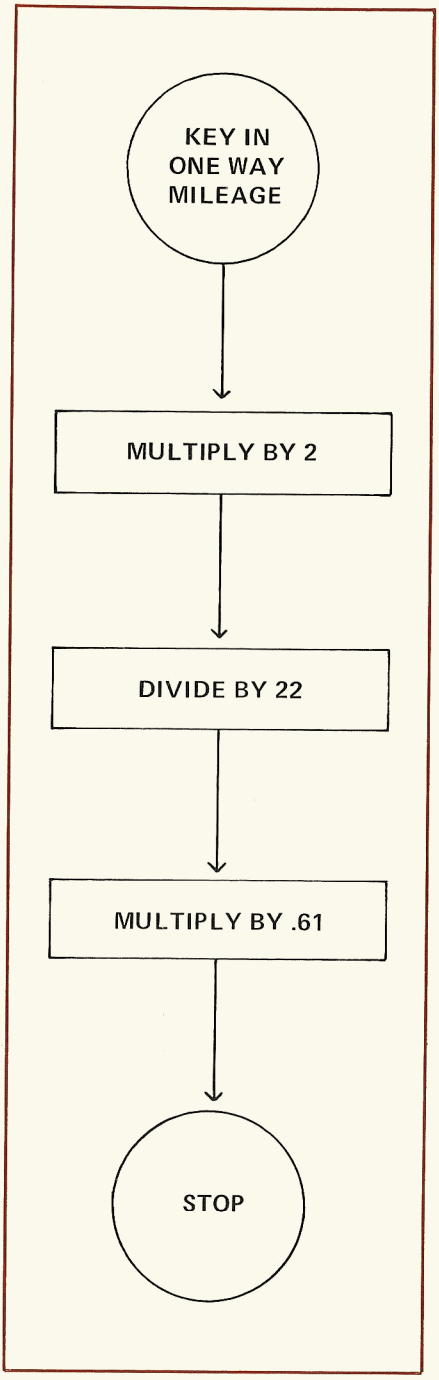
Distance is the variable here, so let's

label distance "d" and complete the transformation of the problem into a mathematical expression:

d x 2 ÷ 22 x .61 = cost ($)

Some programmers use a diagram called a *flow chart* to help them develop programs and to illustrate the operation of a completed program as it is executed by a calculator. To the right you'll see how our gasoline expense problem would appear in flow chart form.

Now that the problem has been converted into a mathematical expression (or a flow chart), we're ready to write the program.



3. Write the program.

A calculator program is nothing more than a listing of the keys you would press to solve the problem manually. Therefore, once you have converted a problem into a mathematical expression, writing the actual program is simple. All you do is list on a note pad or program form the various keys you would press to solve the problem manually.

You must also make provisions for entering variable data into the calculator each time the program is executed. If there is only one variable, as in the case of our gasoline expense problem (distance), arrange your program so the variable is the first number processed by the program. This will allow you to enter the variable and then start program execution. After the program stops to display the answer, you can key in a new variable and execute the same program again. If your program has more than one variable you can program the calculator to stop program execution when it is time to enter each new variable. You can even use spare memory registers to store multiple variables which your program can recall when necessary.

4. Key the program into the calculator.

This is the easiest programming step. Simply switch the calculator to its *program* or *learn* mode, press the sequence of keys listed on your program form, and switch the machine back to its run mode.

5. Test and then run the program.

After you key in the program, the calculator becomes dedicated to solving *your* specific problem. Of course the calculator will only provide accurate results if your program is correct and you keyed it into the calculator without error. Therefore, be sure to test your program by using a variable which gives an easily verified answer.

If the program gives an incorrect answer, you will need to follow the editing instructions given in the next step. If the answer is correct, you can then use the program to solve the problem for which it was written.

In the case of Elmer Travelmore's gasoline expense problem, all you have to do is key in the one-way distance to the first destination (226 miles) and press the *run* or *start* key. In a second or so the display will read 12.53 or $12.53. After you record this result, you can immediately key in the second distance (314 miles) and execute the program again to obtain 17.41. Finally, you can key in the third distance (289 miles) and execute the program a third time to obtain 16.03

6. Edit the program.

If the calculator does not provide a correct answer to a test problem, you will need to edit the program. Better programmable calculators allow you to edit the program for correctness *after* you have keyed it into the machine. All you have to do is set the calculator to the first step in the program and advance the calculator through the program one step at a time.

The display will show a special code for each keystroke entry, and in some cases it will show the step number. If you find an error, some calculators will allow you to overwrite it with the correct keystroke(s). Other models allow you to delete the error and insert a correction.

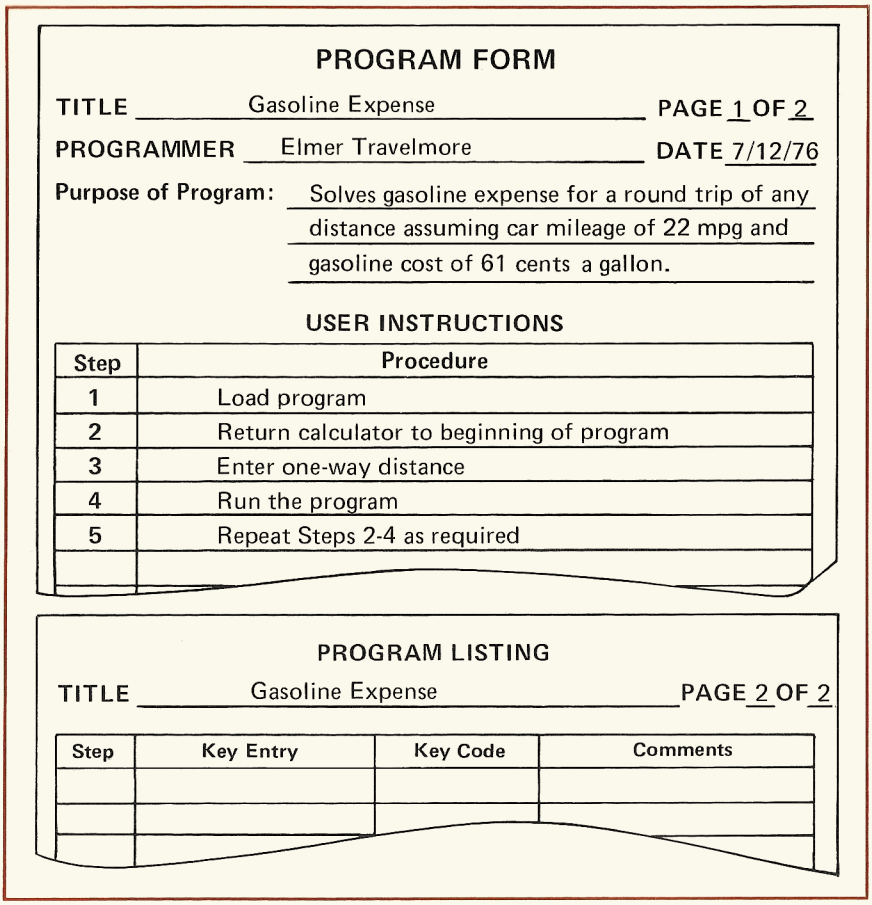
Calculators without editing features require that you reenter the *entire* program if you suspect an error is present. In either case, if the calculator *still* provides incorrect solutions after you have corrected keystroke errors or reentered the program, the program itself may be in error. You will need to return to Step 2 or 3 and try again.

7. Document your program.

One of the most important steps in efficient programming is carefully recording your programs for future use. This step will save considerable time if you need to solve a similar problem in the future. And you'll soon accumulate a library of programs for personal use and exchanging with your friends.

Forms for recording your programs are available from some calculator manufacturers. If you prefer, you can design your own form and use a duplicating machine to make additional copies.

On the following page you will see how the program for the solution of our example problem would be documented on a typical program form.



PROGRAM FORM

TITLE Gasoline Expense PAGE 1 OF 2

PROGRAMMER Elmer Travelmore DATE 7/12/76

Purpose of Program: Solves gasoline expense for a round trip of any distance assuming car mileage of 22 mpg and gasoline cost of 61 cents a gallon.

USER INSTRUCTIONS

Step	Procedure
1	Load program
2	Return calculator to beginning of program
3	Enter one-way distance
4	Run the program
5	Repeat Steps 2-4 as required

PROGRAM LISTING

TITLE Gasoline Expense PAGE 2 OF 2

Step	Key Entry	Key Code	Comments

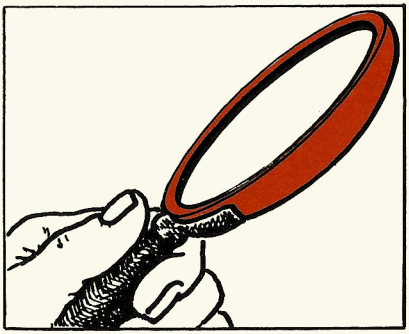
Developing an Effective Programming Style

After you begin writing and using programs you'll soon develop your own personal programming style. Sometimes you will need only seconds to dash off a quick program you've jotted on a note pad or simply worked out in your head! Other times you may follow each of the seven steps listed here. You may even use a flow chart from time to time.

You'll soon learn that there's more than one way to program even the simplest problems. You'll also learn that the shortest way of programming a particular problem is not always apparent at first. And while you might want to invest time condensing a program as an exercise in improving your programming skill, this is not necessary if your first version fits the available memory space.

Developing an effective programming style is a continuing process, but eventually you will accumulate a collection of useful shortcuts and techniques. And, of course, you can always apply the most efficient technique of all by using one of the many programs prepared by experts which are available from some calculator manufacturers.

Programming Features You Should Know About



Programmable calculators incorporate various combinations of programming features, and you should carefully consider several different machines before making a purchase decision. All programmables have at least one control key for starting and stopping program execution. More advanced programmables also have keys which allow you to review and edit your programs. These calculators may also be able to implement such computer-like operations as decision making and transferring from one part of a program to another. Some features greatly enhance the versatility and power of a programmable calculator. The following discussion of programmable calculator features will help you decide how much programming power you require.

Starting and Stopping a Program

You can start or stop program execution on most calculators by pressing one or two program control keys. These keys are variously designated:

START RUN R/S (Run/Stop)	Begins program execution
HALT HLT R/S	Stops program execution
RTN (Return)	Stops program execution and returns calculator to beginning of program memory

Some programmable calculators allow you to execute a program by pressing a *user-definable key* designated:

[A] [B] [C] [D] [E]

You can use these and certain other keys to identify a particular program or program segment by simply preceding the keystrokes you would use to manually solve the problem with this key sequence:

1. Press "label" key
2. Press any user-definable key
3. Key in program

Now your program is "labeled" and you can execute it by simply pressing the appropriate user-definable key.

Labeling is a very useful feature since you can label and store several separate or interactive programs in a calculator simultaneously and select any of them by simply pressing the appropriate user-definable key. And, as we shall soon see, labeling also permits many important programming operations.

Program Editing

The utility of a programmable calculator is greatly influenced by the versatility of its editing features. As mentioned earlier, calculators without editing features require you to reenter complete programs which have only minor mistakes or errors. But an efficient cluster of editing keys will allow you to change instructions, delete mistakes and generally "debug" a program in minutes.

Here are some of the editing features which are available in various combinations on better programmable calculators:

[SST] Single step (advances program one step; in [RUN] mode some calculators actually *execute* the program one step at a time as [SST] is pressed to permit intermediate results to be reviewed)

[BST] Backstep (moves program back one step)

[GTO] GO TO (directs calculator to proceed to any specified program *or* program step; e.g. GTO A or GTO 014)

[DEL] Deletes any specified instruction from program memory; all subsequent instructions are moved up one step

Transferring Capabilities

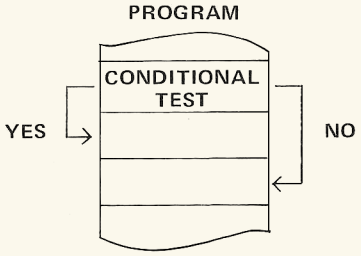
Calculators normally sequence through programs step-by-step (1... 2...3...). Some programmables can alter this sequential program execution and transfer to another part of a program—and possibly back again. This capability provides immense programming power.

Program transfers are made possible by assigning *addresses* to various steps or segments of a program. Some calculators can use the line number of each step in the program as an address (e.g. a typical program instruction might be "GO TO 136"). More sophisticated calculators use labels for transfer addresses (e.g. "GO TO A"). This greatly simplifies program writing and editing since you don't have to keep track of address changes each time you revise the program. Program transfers can be executed either unconditionally or if a specified condition is first met. Both types of transfers are described here.

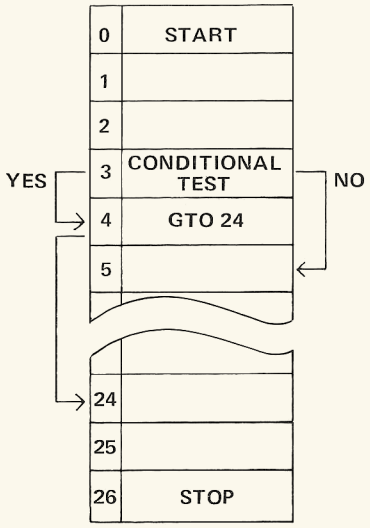
Unconditional Transfer Instructions. We noted that when pressed from the keyboard the [GTO] key found on some programmables will cause the calculator to proceed to a specified program step or label. You can also include [GTO] as part of a program. When the calculator

reaches the [GTO] instruction it will automatically transfer program execution to the specified location. This feature is very valuable for cycling the calculator through a section of program more than once.

Conditional Transfer Instructions. Another important type of transfer is the "conditional transfer." Conditional transfer instructions give calculators a powerful decision-making capability. More than a dozen conditional transfer instructions are available in various combinations on some programmable calculators, and most of them ask the question: "Does the number in the display meet a specified condition?" If *yes*, the calculator continues normal program execution. If *no*, the calculator skips or jumps one or more steps and then continues normal execution. Here's how this procedure looks pictorially:



Conditional transfer instructions are usually paired with unconditional [GTO] instructions. Here's a typical example:



Conditional transfer operations are very useful since they give your program the ability to respond automatically to a variety of circumstances. They can even be used to repetitively cycle or loop the calculator through a segment of program until a desired result is reached.

Below are some of the specific conditional transfer instruction keys available in various combinations on sophisticated programmables:

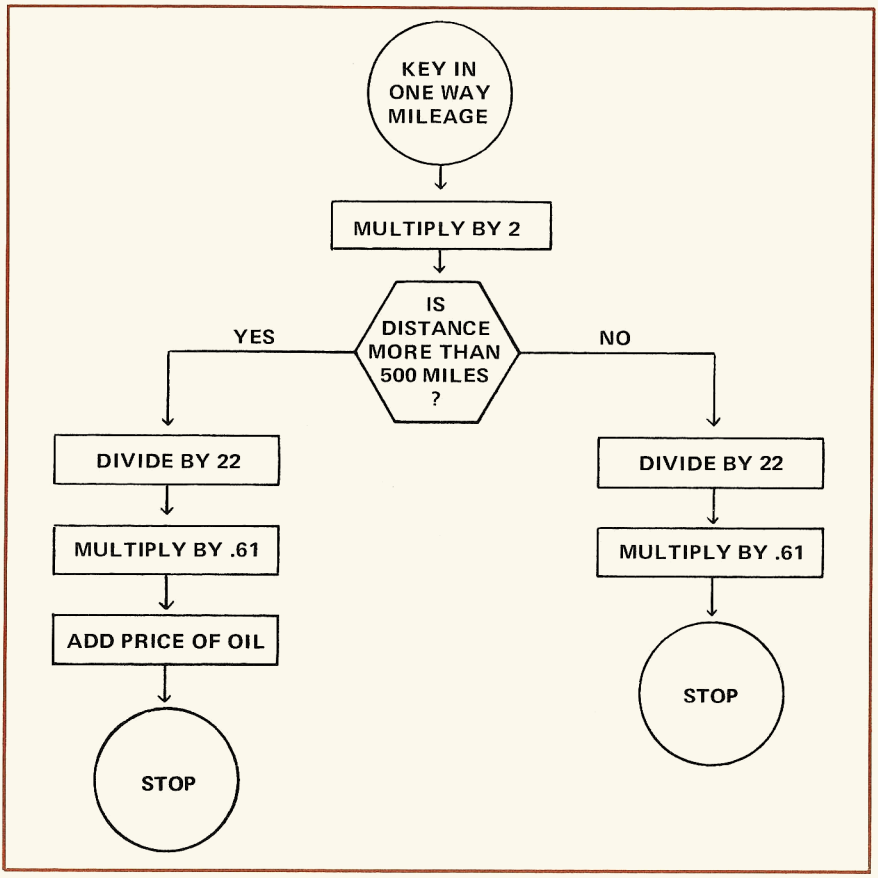
[X<0] or [if neg] [X>0] or [if pos] [X≠0] [if not zero] [X=0] or [if zero]	Compares value in display (X) with zero. If test is true, program execution continues sequentially. If test is false, program skips one or more steps.
[X<Y] [X>Y] [X≠Y] [X=Y]	Compares value in display (X) with value in Y register. If test is true, program execution continues sequentially. If test is false, program skips one or more steps.
[if err]	If error signal is in display, program execution continues sequentially. If not, program skips one or more steps.

The best way to see how these conditional tests work is to apply one in an actual example. Returning to the gasoline expense problem given earlier. let's assume Elmer Travelmore's roadster uses a quart of oil every 500 miles. Since Elmer wants to plan his vacation expenses to the nearest penny, how can he rewrite his gasoline expense program to reflect oil expense?

One way is to include a conditional test which asks if the total mileage exceeds 500 miles. If so, the price of a quart of oil is added to the total gasoline cost. If not, the program continues normal execution. Here's how Elmer's program would now look in flow chart form:

Flags and Counters: Additional Conditional Transfer Capability

The most advanced programmable calculators may also have such specialized conditional transfer instructions as flags and counters. Flags are simply status signals which can be set or cleared by you or the calculator. When the calculator encounters a flag, it asks whether or not the flag is set. If true, the program continues sequentially; if not, the program skips one or more steps.



There are various designations for flag keys, including:

SF	or	ST FLG	Set flag
CF			Clear flag
F?	or	if FLG	Is flag set?

Counters cycle the calculator through a segment of program any number of times you desire by automatically adding or subtracting *one* from the number stored in a particular memory register. When the value in the memory register is *not* zero, program execution continues sequentially. When the value in the register reaches zero, the program skips one or more steps and resumes execution. The keys for these two functions are normally designated:

ISZ Increment and Skip on Zero

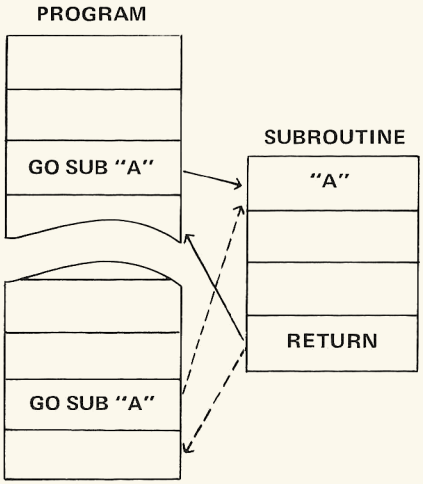
DSZ Decrement and Skip on Zero

Subroutines: Increased Program Efficiency

Frequently solutions to mathematical problems use an identical formula or equation two or more times within the solution. Assuming your program does not exceed the program memory of your calculator, you can always repeat the set of instructions each time it is needed.

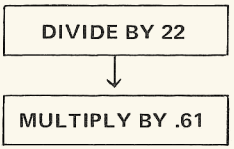
A much more efficient method is to program the repeated section once as a separate program called a *subroutine* and assign it a label with a user-definable key. Program execution will then divert to the subroutine whenever the appropriate label is encountered. A special instruction designated RTN is placed at the end of a subroutine to return program execution to the main program after the subroutine has been executed.

Here's how a subroutine looks pictorially:

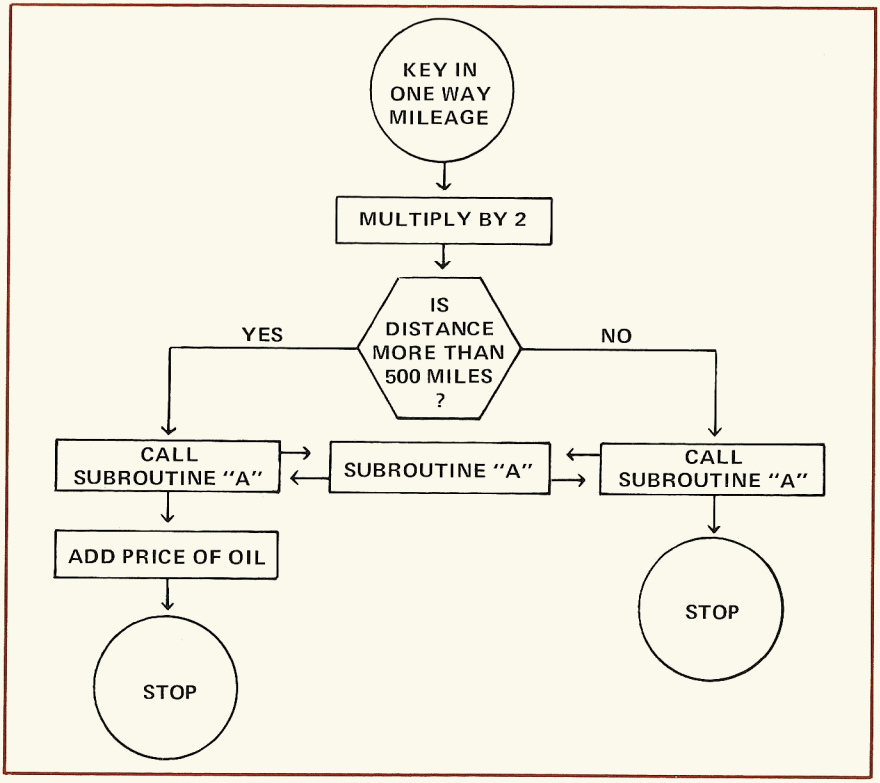


We can return to Elmer Travelmore's gasoline expense program to see a practical application of a subroutine. Let's assume Elmer is as frugal with the memory space in his calculator as with his vacation budget. If Elmer's calculator is an advanced model with a subroutine capability and a row of user-definable keys, how can he simplify his program?

Looking back at the latest version of Elmer's program (p. 10) we see that one instruction sequence is repeated *twice*:



Elmer can easily label this program segment with one of the user-definable keys on his calculator by pressing LBL and A . He can then tag a RTN instruction on the end of the segment. Now he has a subroutine which his program can call into action



whenever it is required. In the above diagram you'll see how the new version of the program would look in flow chart form.

Of course this is a very simple example of a subroutine. But it does illustrate the efficiency this important capability adds to programmable calculators.

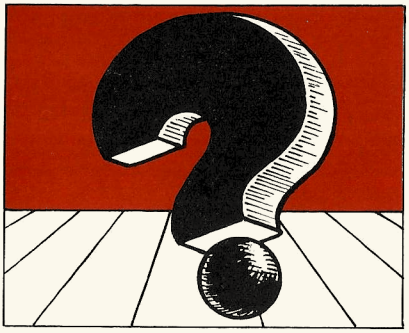
Indirect Control

One of the most powerful of the computer-like features of sophisticated programmable calculators is *indirect control*. Depending upon the calculator, this amazingly flexible feature allows you to do such things as specify which storage register is to be addressed by STORE and RECALL instructions according to the number stored in a special indirect control register. This feature permits the calculator to *automatically* store a series of results in successive memory registers.

Indirect control also allows the calculator to transfer to any program line number stored in the indirect control register. Other forms of indirect control allow the calculator to transfer program execution to a label stored in the indirect control register, adjust the display format, and increment, decrement, or perform arithmetic upon the contents of any memory designated by the indirect control register.

These various operations may not seem abundantly clear to you now, especially if you are new to programming. But you will soon find countless applications for indirect control if you select a calculator with this advanced feature.

Choosing a Programmable Calculator: What You Need To Know



Selecting a programmable calculator is in some ways easier than choosing a preprogrammed model since there are fewer machines from which to choose. Nevertheless, you should carefully evaluate the programming features of the various machines in your price range before making a purchase decision.

Programmable calculators can be classified according to their programming power as elementary, intermediate or fully programmable.

1. Elementary Programmables—These machines range from basic four-function calculators with a short program memory to 100-step scientific machines. While some of these calculators can handle relatively difficult problems, the limited number of program instructions and the lack of editing features slow down problem solving and make complex problem solving difficult or impossible.

2. Intermediate Programmables—Calculators in this class have from 49 to 100 program steps, multiple storage registers and a wide range of preprogrammed functions. They also have a "pause" feature which automatically stops program execution for one second to permit you to read out intermediate data.

An important feature recently introduced to the intermediate programmable is a *continuous* program memory. New semiconductor technology has resulted in a memory which consumes so little power that it can be left on even when the rest of the calculator is turned off. This handy feature can save considerable time if you frequently use a favorite program, since calculators without a continuous memory must be reprogrammed each time they are turned on.

3. Fully Programmables—In addition to all the programming features of intermediate machines, calculators in this class have such computer-like features as subroutine capability, user-definable keys, flags and indirect program control. Best of all, they can also record programs on tiny magnetic cards no larger than a stick of chewing gum. This remarkable feature means you can *permanently* record your programs and then load them into the calculator in only seconds.

The newest calculators in this class incorporate card readers which can perform such operations as recording and loading *data*, as well as programs, on magnetic cards. These card readers can accept programs or data from either side of a card, no matter which is read first. They even flash a signal in the display if both sides of a card must be read.

With amazing features like these, it's little wonder why fully program-

mable calculators are often called *pocket computers.*

How To Evaluate Program Capacity

Be wary of manufacturers' claims when evaluating a calculator's program capacity. The total number of available program steps is certainly important, but the efficiency with which those steps can be used is even more significant.

A calculator's language plays a limited role in determining program capacity. RPN makes somewhat more efficient use of program memory space since RPN programs are generally shorter than those of calculators which use one of the algebraic languages. This is because RPN usually permits problems to be solved with somewhat fewer keystrokes. Also, the operational stack used by RPN calculators permits numerous programming shortcuts not possible on algebraic models.

Even more important is whether or not the calculator has fully merged keycodes. All intermediate and fully programmable machines use one or more prefix keys to designate the secondary functions assigned to various keys. The resultant two-step sequence is usually merged into a single step of program memory in what is known as a *merged prefix.*

Other multiple keystroke operations may *not* be merged on some calculators. For example, adding a number to memory register two requires three steps of program memory on a non-merged machine:

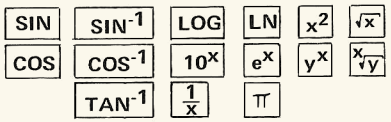
1	STO
2	+
3	2

A fully merged machine squeezes all three keystrokes into one step of program memory:

1	STO + 2

The Importance of Preprogrammed Functions

Since you will often use a programmable calculator in its *non*programmed mode, you should consider the relative importance of the various keyboard functions when making a purchase decision. Most intermediate and fully programmable calculators include such common scientific functions as these:



Intermediate and fully programmables may also have keys for truncating the integer or fractional part of a number (INT and FRAC respectively), displaying the absolute value of a number (ABS), polar-to-rectangular coordinate conversions (P or P/R and R or R/P), and conversion of hours-minutes-seconds to decimal format (H.MS or D.MS and →H or D.MS-1).

For more complete explanation of these and other preprogrammed functions, obtain a copy of "What to look for before you buy an advanced calculator" from your calculator dealer.

Calculator Construction

As you can see, there is a wide diversity in the programming power of various programmable calculators. There is also considerable variation in the physical design and human engineering of available machines. You should give close attention to both these factors when selecting a calculator for purchase. Here is a listing of some of the more important design features:

• The Display—The display should be legible, provide a high contrast with its background and have an adequate viewing angle. Scientific nota-

tion is essential so that very small and very large numbers can be displayed.

• The Keyboard—You should select a calculator with snap-action keys. These keys are more costly than conventional keys, but their tactile feedback verifies entries. Also, snap-action keys are less likely to cause unintentional double entries than ordinary keys.

The keys should be clearly labeled, preferably with a double injection-molded symbol which will not wear off with use. Finally, the keyboard should be well organized and visually appealing. Programmable calculators have far more instructions, operations and functions than preprogrammed models, so a well-planned keyboard layout will minimize confusion and reduce "hunt-and-peck" keystroking.

• The Quality of Construction—Finally, assess the overall construction quality of the various calculators you are considering. Are there openings where dirt and other contaminants can enter the case? Is there a moisture barrier under the keyboard to keep out spilled liquids? Are batteries easy to replace? Is the battery compartment sealed or does removing the batteries expose the circuit board? Is the calculator compact enough for your purposes?

Manufacturer's Support

Programmable calculators are manufactured by a number of firms, but only a few companies supply well-documented programs which can be used with their machines. You will eventually accumulate your own personal library of programs, particularly if you are an adept programmer. But if you are a beginner you should select a calculator backed by a good selection of well-documented programs.

Program development is a costly procedure, and you can generally expect to pay more for calculators which are supplied with program libraries. But the small additional price is a real bargain when compared to the many hours required to produce even a limited set of personal programs. Also, programs prepared by experts in fields other than your own will let you solve problems in a wide range of specialized disciplines—and with no prior training.

A Final Note

Hopefully this booklet has answered many of your questions about pocket programmable calculators. An even better way to appreciate the amazing power and versatility of these remarkable machines is to ask for a demonstration of one or more models at your calculator dealer. Then, if you feel programmability will expand your problem-solving ability, carefully compare the computational capabilities and physical construction of competing machines before making a purchase decision.

At first, writing your own programs may seem more difficult than it actually is. But in time you will find program writing to be a natural and even enjoyable part of solving a problem. You will also find that a well-designed programmable calculator supported by a versatile selection of program instruction keys will enhance your problem-solving ability for years to come.

Introducing Hewlett-Packard: The first family of programmable calculators.

Today, when you choose a Hewlett-Packard programmable calculator, you will be choosing one of the best pocket calculators money can buy. *Any* amount of money.

Hewlett-Packard was the first company to introduce the advanced pocket calculator—back in 1972. The first company to introduce a fully programmable pocket calculator. And the first company to introduce advanced pocket calculators accepted by the business and scientific communities.

The First Family is the choice of professionals.

Hewlett-Packard calculators are recognized by more than one million owners as The First Family worldwide. Including Nobel Prize winners. USA-USSR astronauts. Explorers. Educators. Scientists. Business people. And students. Maybe you.

Easy to use.

Despite its sophistication, you'll find an HP calculator easily handles the kinds of programmable problems you'll face. A short review of the owner's instruction manual should make you comfortable with any model in a matter of hours.

Make your first calculator your last.

Any good calculator is going to cost some money. So be sure to get one that will meet all your needs now and in years to come. A calculator that will last, from a company that will stand by you. Hewlett-Packard makes that kind of a calculator and is that kind of company. Ask around.

For additional details plus the name of your nearest dealer, simply write: Hewlett-Packard, Dept. 226A, 19310 Pruneridge Avenue, Cupertino, CA 95014. After Nov. 1, 1976 write: Hewlett-Packard, Dept. 226A, 1000 N.E. Circle Boulevard, Corvallis, Oregon 97330.

HEWLETT hp PACKARD

Sales and service from 172 offices in 65 countries.
Dept. 226A, 19310 Pruneridge Avenue, Cupertino, CA 95014.

616/27

HP-25C

Scientific Programmable with Continuous Memory.

The HP-25C is our keystroke programmable. It can solve automatically the repetitive problems every scientist and engineer faces. What's more, its Continuous Memory capability lets you retain programs and data even when it's turned off.

- Continuous memory capability.
- 72 built-in functions and operations.
- Keystroke programmability.
- Branching, conditional test and full editing capability.
- Eight addressable memories.
- We also offer the HP-25, without the Continuous Memory feature.

5952-6067D Printed in U.S.A. 8/76

HP-67

Powerful, fully programmable.

The HP-67 is the most powerful, fully programmable scientific pocket-sized calculator Hewlett-Packard's ever built. And it's easy to use, too. Improper card loading is virtually impossible with our "smart" card reader. Plus it's exceptionally easy to edit. If you need a powerful pocket-sized fully programmable, this is it.

- Handles programs of up to 224 steps.
- All prefix functions are merged.
- Directly records contents of all 26 data storage registers onto magnetic cards.
- 3 types of addressing—Label, Relative and Indirect.
- Easy to use—"smart" card reader automatically records program status of flags, display, angular modes.
- Standard Application Pac with 15 programs of general appeal included free. Optional Pacs in Engineering, Finance, etc., available at modest cost.
- We also offer the brief-case portable HP-97, featuring all the power of the HP-67 plus a built-in thermal printer.